16-519

Index Partitioning Based on Document Relevance for Document Indexes

Technical Field

The invention pertains generally to the field of document indexing for use by internet search engines and in particular to an index scheme that partitions an index based on a projected relevance of documents.

Background of the Invention

Typical document indexing systems have word occurrence data arranged in an inverted content index partitioned by document. The data is distributed over multiple computer systems that are dedicated to index storage with each computer system handling a subset of the total set of documents that are indexed. This allows for a word search query to be presented to a number of computer systems at once with each computer system processing the query with respect to the documents that are handled by the computer system.

An inverted word location index partitioned by document is generally more efficient than an index partitioned by word. This is because partitioning by word becomes expensive when it is necessary to rank hits over multiple words. Large amounts of information are exchanged between computer systems for words with many occurrences. Therefore, typical document index systems are partitioned by document and queries on the indexed documents are processed against the contents of the indexes until a sufficient results set is obtained. While the number of documents indexed in search engines is growing, in many cases the results for most queries come from a small

portion of the entire set of documents. Therefore it may be inefficient to search indexes that contain documents that are less likely to return results in response to a query.

Summary of the Invention

Storing and partitioning documents in an index based on relevance enables a search of the index to be terminated when sufficient results have been found without requiring a scan of the entire index.

A static rank is assigned to each document that captures a relative relevance of the document. The documents are ordered and partitioned based on the static rank of the documents. The ordered and partitioned documents are indexed by mapping a document location to words contained in the document. In response to a query the index partitions are searched in static rank order from highest to lowest. After each partition is searched a score is calculated based on the results returned so far and the static rank of the next partition. The calculated score is compared to a target score and no further partitions are searched once the calculated score exceeds the target score.

The static rank may be based on a number of links that refer to the document. Alternatively the static rank may be based on a number of times a document is accessed by previous queries. A dynamic rank can be used to quantify the quality of results returned so far. The dynamic rank may be based on a number of documents returned so far or a measure of the relevance of the documents returned so far.

Brief Description of the Drawings

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which:

Figure 1 illustrates an exemplary operating environment for a system for processing and routing database queries;

Figure 2 is a block diagram of a computer system architecture for practicing an embodiment of the present invention;

Figure 3 is a functional block diagram of an index generation process that can be used in practice of an embodiment of the present invention;

Figure 4 is functional block diagram of an index serving process that can be used in practice of an embodiment of the present invention; and

Figure 5 is a flowchart of a method for practicing an embodiment of the present invention.

Detailed Description of the Preferred Embodiments

Figure 2 illustrates a block diagram of a search engine 10 that features a document index system that takes in document data and indexes the content of the documents by word. A web crawler 235 accesses documents on the internet to be indexed by the index system and passes the document data to an index builder 240 that parses the document and extracts words and word locations for storage in index serving rows 250. The web crawler, index builder, maintenance of the index serving rows as well as the search engine are typically constructed in software executing on a computer system 20 (Figure 1). The computer system 20 is in turn coupled by means of communications connections to other computer systems by means of a network.

The index serving rows 250 can be constructed as a matrix of computer systems 20 with each computer system in a row storing word locations for a subset of the documents that have been indexed. Additional rows of computer systems 20 in the index serving rows may store copies of the data that is found in computer systems in the first row to allow for parallel processing of queries and back up in the event of computer system failure. The index information stored in each computer system 20 in the index serving rows is partitioned by the index builder 240. The size of the partition may range from a size that advantageously fits into cache memory to a single partition per computer system.

Index Partitioning Based on Document Relevance

As discussed in the background, partitioning by document is a typical way of constructing document indexes. While the number of documents indexed in search engines is growing, in many cases the results for most queries come from a small portion of the entire set of documents. In indexing schemes where documents are indexed without regard to the projected relevance of the documents much effort and expense will be wasted searching documents that have historically yielded few results.

Referring again to Figure 2, a computer system layout architecture 10 for a document search system is shown. Auto pilot computer systems 215 coordinate the working of the other computer systems in the system as it processes user queries and requests. A rank calculation module 245 tracks the popularity of web sites and feeds this information to a web crawler 235 that retrieves documents from the internet based on links that exist on web pages that have been processed. An index builder 240 indexes the words that are found in the documents retrieved by the crawler 235 and passes the data to

a set of index serving rows 250 that store the indexed information. In the embodiment described here, the index serving rows include ten "rows" or sets of five hundred computer systems in each row. Indexed documents are distributed across the five hundred computer systems in a row. The ten rows contain the same index data and are copies of one another to allow for parallel processing of requests and for back up purposes.

A front end processor 220 accepts user requests or queries and passes queries to a federation and caching service 230 that routes the query to appropriate external data sources as well as accessing the index serving rows 250 to search internally stored information. The query results are provided to the front end processor 220 by the federation and caching service 230 and the front end processor 220 interfaces with the user to provide ranked results in an appropriate format. The front end processor 220 also tracks the relevance of the provided results by monitoring, among other things, which of the results are selected by the user.

Figure 3 shows a functional block diagram that provides more detail on the functioning of the web crawler 235, index builder 240, and index serving rows 250. The crawler includes a fetcher component 236 that fetches documents from the web and provides the documents to be indexed to the index builder 240. Information about URLs found in the indexed documents 261 is fed to the crawler 235 to provide the fetcher 236 with new sites to visit. The crawler may use rank information from the rank calculation module 245 to prioritize the sites it accesses to retrieve documents.

Documents to be indexed are passed from the crawler 235 to the index builder 240 that includes a parser 265 that parses the documents and extracts features from the

documents. A link map 278 that includes any links found in a document are passed to the rank calculating module 245. The rank calculating block 245 assigns a query-independent static rank to the document being parsed. This query-independent static rank can be based on a number of other documents that have links to the document, usage data for the URL being analyzed, or a static analysis of the document, or any combination of these or other factors.

Document content, any links found in the document, and the document's static rank are passed to a document partitioning module 272 that distributes the indexed document content amongst the computer systems in the index serving row by passing an in memory index 276 to a selected computer system. A link map 278 is provided to the rank calculation module 245 for use in calculating the static rank of future documents.

The documents whose index in each computer system is partitioned based on the static rank assigned to documents in the partition with documents of highest rank being located in partitions that are first accessed in response to a query. One or more thresholds are defined for the different partitions. These thresholds are based on estimates of what percentage of queries would likely be answered by documents in each range and by the size of each partition. It may be advantageous to place fewer documents in the higher ranked partitions. For example, documents having static rank values between 91-100 may go in a first partition, rank values of 51-90 in a second partition, and 1-50 in a third partition.

Partitioning provides multiple advantages. Less processing occurs since the queries are run on the smaller sets of documents first. I/O costs can be reduced since the smaller higher ranked partitions will get more queries and have greater cache locality.

I/O operations can be eliminated for the highest ranked partitions if their data can be loaded into memory and this approach takes advantage of more efficient operating system memory management. Large page allocations can be used so that less CPU page table space is necessary and CPU cache efficiency is improved.

Figure 4 illustrates a functional block diagram for the handling of search queries with respect to the index serving rows 250. The search query is routed to a query request handler 123 that directs the query to the federation and caching service 230 where preprocessing 131 is performed on the query to get it in better condition for presentation to a federation module 134 that selectively routes the query to data sources such as a search provider 137 and external federation providers 139. The search provider 137 is an "internal" provider that is maintained by the same provider as the search engine. External federation providers 139 are maintained separately and may be accessed by the search engine under an agreement with the search engine provider. To evaluate a query on the search provider 137, the search provider routes the query 141 to a query fan out and aggregation module 151 that distributes the query over the computer systems in a selected row of the index serving rows 250 and aggregates the results returned from the various computer systems. The index query 155 from the fan out module is executed on the first index partition in the computer system 157A that has the highest document static rank. The index query may also be executed in turn on successive index partitions 157A....157N of lower ranking documents depending on several factors to be discussed below.

Figure 5 is a flowchart illustrating a method 500 that can be used to determine whether it is necessary to search a next index partition or whether the results returned by

accessing the previous index partition are sufficient to answer the query. This determination is made based on two factors: the static rank of documents in the next partition and a dynamic rank that characterizes the quality of the results set returned so far. The dynamic rank can be determined by simply counting the number of documents that have been found that answer the query. More complex approaches to computing a dynamic rank include a measure of the quality of the matching documents found thus far such as the prominence of the search word in the document or a number of times a search word is present in the document.

A counter N is initialized in step 510 for pointing to the correct index partition to be scanned. In step 520, a target score TS and tuning factor α are input. The target score relates to the quality of results that must be obtained before the scanning of index partitions is stopped. For example, a target score of 100 would mean that scanning would not be discontinued until it is determined that no more results will be found in the next partition. The tuning factor α determines the weight given to the static rank of the documents in the next partition relative to the weight given to the dynamic rank.

In step 530, a partition is scanned and results or hits are recorded. A hit occurs when a document containing a query search word is detected in the partition. A dynamic rank is calculated in step 535 based on the quality of the hits recorded so far using some quality metric as discussed above. In steps 540 and 550, a score is calculated for the results generated so far by applying the tuning factor α and its complement to the static rank of the next partition and dynamic rank of the results obtained so far, respectively. If the calculated score is higher than the target score, the next partition is scanned in steps

560 through 550 and if the calculated score is below the target score, the search process is halted and the results are returned, step 570.

The higher the portion of the ranking score that is derived from the static rank the less likely the next partition will be scanned and results will be obtained more quickly but the relevance of the results returned may be compromised. For example, a target score is set to 100 and α is set to "1" so that ranking score is based solely on the static rank of the next partition. If ten results is the number of results that are to be returned by a query, and ten results are found in the first partition scanned, regardless of quality of the results obtained so far. By comparison if the tuning factor α is set to "0.5" and the quality of the hits obtained so far is relatively low, then the next partition would be scanned because of the influence of the low dynamic rank in the equation. Another way to tune the scanning process is to lower the target score which will make it less likely that the next partition is scanned, but possibly at the price of less relevant results. The tuning factor α can be changed as a factor of load to relieve load from the system by halting scans earlier.

Stopping the scanning of indexes early has many advantages. Less memory is needed to cache the index since more queries are satisfied form a smaller set of the index. Because the most relevant documents are stored together, the portion of the index that is cached can be used to answer more queries. In some cases it may be possible to force the data for some partitions to fit completely in memory. Less memory can also be translated into less disk I/O operations and by only querying a smaller portion of the index also saves CPU time. Reading and ranking can take significant CPU resources and stopping the query early could save processing time.

Exemplary Operating Environment

Figure 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Figure 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including system memory 22 to processing unit 21. System bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. System memory 22 includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system (BIOS) 26, containing the basic routines that help to transfer information between elements within personal computer 20,

such as during start-up, is stored in ROM 24. Personal computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29 and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD ROM or other optical media. Hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by computer, such as random access memories (RAMs), read only memories (ROMs), and the like may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 129, optical disk 31, ROM 24 or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A database system 55 may also be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25. A user may enter commands and information into personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to processing unit 21 through a serial port interface 46 that is coupled to system bus 23, but may be connected by other interfaces,

such as a parallel port, game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices such as speakers and printers.

Personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. Remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to personal computer 20, although only a memory storage device 50 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When using a LAN networking environment, personal computer 20 is connected to local network 51 through a network interface or adapter 53. When used in a WAN networking environment, personal computer 20 typically includes a modem 54 or other means for establishing communication over wide area network 52, such as the Internet. Modem 54, which may be internal or external, is connected to system bus 23 via serial port interface 46. In a networked environment, program modules depicted relative to personal computer 20, or portions thereof, may be stored in remote memory storage device 50. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

It can be seen from the foregoing description that partitioning a document index based on document relevance and scanning only as much of the index as is necessary to obtain sufficient results saves memory and processing resources. Although the present invention has been described with a degree of particularity, it is the intent that the invention include all modifications and alterations from the disclosed design falling within the spirit or scope of the appended claims.